MODEL EVALUATION

1. **Logistic Regression**
   A linear model used for binary classification, making predictions based on the weighted sum of input features. It performs well when the relationship between features and target is linear but struggles with complex data**.**
   - Accuracy: 0.498
   - Precision/Recall/F1-score
     - ✓ Positive class (1): High recall (0.96) but poor precision (0.50)
     - ✓ Negative class (0): Very low precision (0.45) and recall (0.03
   - Performance Summary
     - ✓ Model leans heavily towards predicting the positive class.
     - ✓ Poor at handling class imbalance or capturing the negative sentiment accurately.
2. **Naive Bayes**
   A probabilistic classifier based on Bayes' theorem, assuming feature independence. It is fast and effective for text classification but can be limited by this strong independence assumption.
   - Accuracy: 0.510
   - Precision/Recall/F1-score
     - ✓ Positive class (1): Decent precision (0.51) and recall (0.76).
     - ✓ Negative class (0): Poor recall (0.26).
   - Performance Summary
     - ✓ Performs marginally better than Logistic Regression.
     - ✓ It's better at recognizing positive sentiment but still struggles with negative class detection.
3. **Support Vector Machine (SVM)**
   A robust model that finds the best hyperplane to separate data points, suitable for high-dimensional data but slow with large datasets.
   - Accuracy: 0.498
   - Precision/Recall/F1-score
     - ✓ Positive class (1): High recall (0.96) but poor precision (0.50).
     - ✓ Negative class (0): Precision (0.46), recall (0.03) are very low.
   - Performance Summary
     - ✓ Similar to Logistic Regression, biased towards predicting the positive class.
     - ✓ Struggles with negative sentiment.
4. **Random Forest**
   An ensemble model that builds multiple decision trees and outputs the majority class. It handles non-linear relationships well but may be overfit without proper tuning.
   - Accuracy: 0.616
   - Precision/Recall/F1-score
     - ✓ Better balance across both classes (e.g., 0.60 precision for negative class, 0.63 for positive).

    - ✓ Still struggles with class imbalance, with some decrease in recall for the positive class.
- Performance Summary
    - ✓ Performs better than Naive Bayes, Logistic Regression, and SVM.
    - ✓ Still not ideal but captures both positive and negative sentiments reasonably well.

5. LSTM (Long Short-Term Memory)

A deep learning model designed to learn dependencies in sequential data, particularly effective for text-based tasks like sentiment analysis, though it requires more training time and resources.

- Accuracy: 0.775 (Validation accuracy: 0.788).
- **Loss:** 0.465 (Training) / 0.447 (Validation).
- Performance Summary
    - ✓ Best performing model with an accuracy of 77.5%.
    - ✓ The validation accuracy of 78.8% indicates strong generalization to unseen data.
    - ✓ LSTM, being a deep learning model, excels at capturing long-term dependencies in text, making it well-suited for sentiment analysis.

Overall Comparison

Model	**Accuracy**	**Strengths**	**Weaknesses**
Logistic Regression	0.498	Simple to implement	Poor handling of negative sentiment
Naive Bayes	0.51	Good for text data	Struggles with negative class
SVM	0.498	Works well for binary classification	Biased towards positive class
Random Forest	0.616	Captures both classes better	Still suffers from some imbalance
LSTM	0.775	Captures complex text dependencies	Requires more computational resources

Conclusion: LSTM is the Best Model

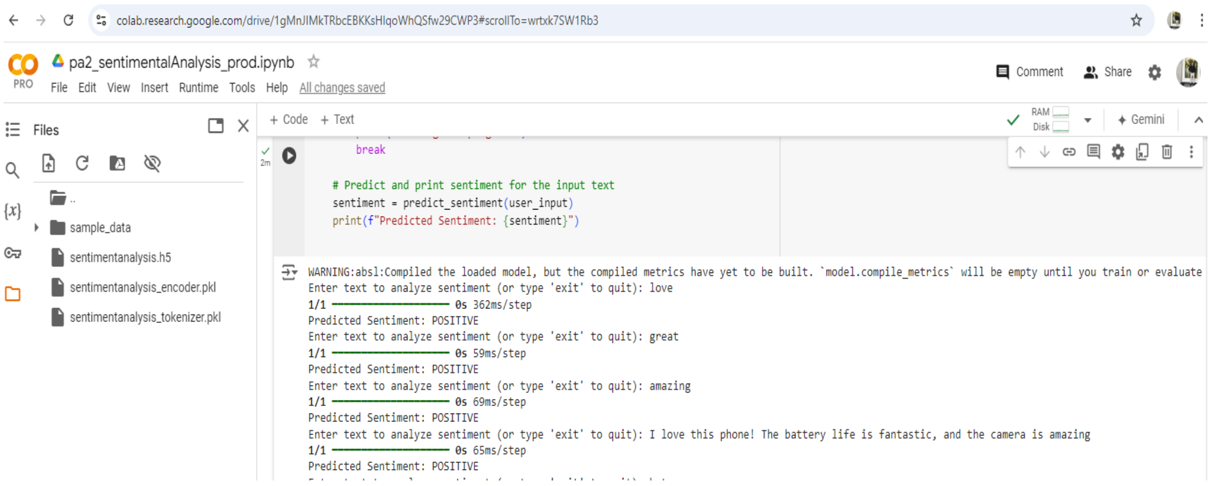
LSTM outperforms all the other models, achieving a significantly higher accuracy of 77.5%. Its ability to model sequential data and understand long-term dependencies makes it ideal for sentiment analysis. While traditional ML models like Logistic Regression and SVM struggle with class imbalance and context, LSTM handles these challenges effectively**.** Hence, Prediction and Inference Module is based on LSTM model.

TESTING

Note: Recommend running test on google Colab.

Sample output screenshots.

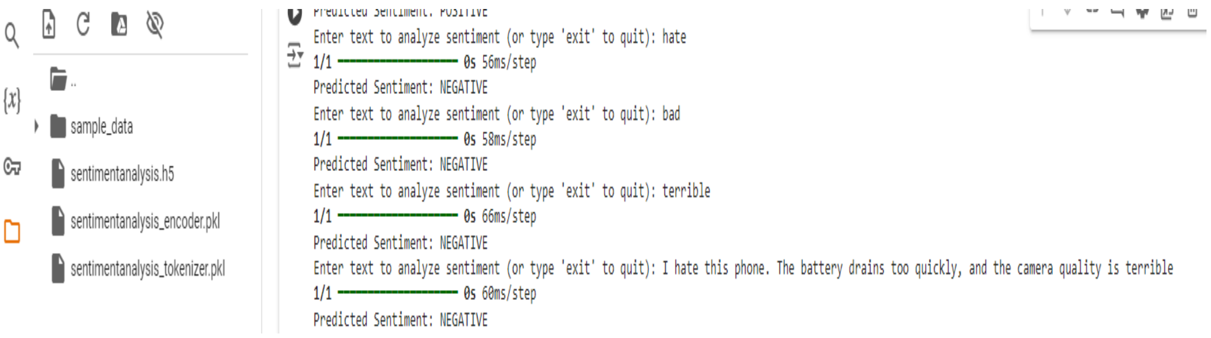
a. **Positive Tweets Scenario**


colab.research.google.com/drive/1gMnJIMkTRbcEBKKsHlqoWhQSfw29CWP3#scrollTo=wrtxk7SW1Rb3
pa2_sentimentalAnalysis_prod.ipynb
PRO
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Gemini
2m
..
sample_data
sentimentanalysis.h5
sentimentanalysis_encoder.pkl
sentimentanalysis_tokenizer.pkl

```
        break

    # Predict and print sentiment for the input text
    sentiment = predict_sentiment(user_input)
    print(f"Predicted Sentiment: {sentiment}")
```

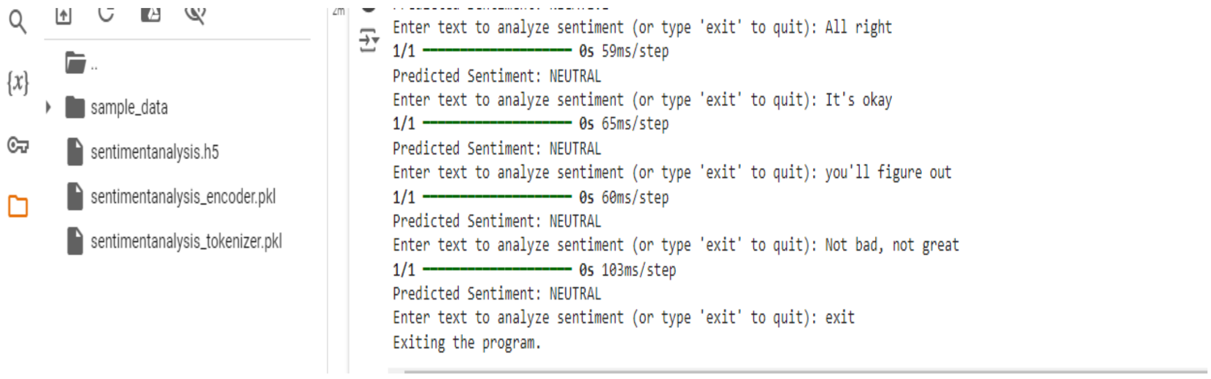
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate
Enter text to analyze sentiment (or type 'exit' to quit): love
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 362ms/step
Predicted Sentiment: POSITIVE
Enter text to analyze sentiment (or type 'exit' to quit): great
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
Predicted Sentiment: POSITIVE
Enter text to analyze sentiment (or type 'exit' to quit): amazing
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step
Predicted Sentiment: POSITIVE
Enter text to analyze sentiment (or type 'exit' to quit): I love this phone! The battery life is fantastic, and the camera is amazing
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
Predicted Sentiment: POSITIVE
```

b. **Negative Tweets Scenario**


..
sample_data
sentimentanalysis.h5
sentimentanalysis_encoder.pkl
sentimentanalysis_tokenizer.pkl

```
Predicted Sentiment: POSITIVE
Enter text to analyze sentiment (or type 'exit' to quit): hate
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
Predicted Sentiment: NEGATIVE
Enter text to analyze sentiment (or type 'exit' to quit): bad
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
Predicted Sentiment: NEGATIVE
Enter text to analyze sentiment (or type 'exit' to quit): terrible
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step
Predicted Sentiment: NEGATIVE
Enter text to analyze sentiment (or type 'exit' to quit): I hate this phone. The battery drains too quickly, and the camera quality is terrible
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
Predicted Sentiment: NEGATIVE
```

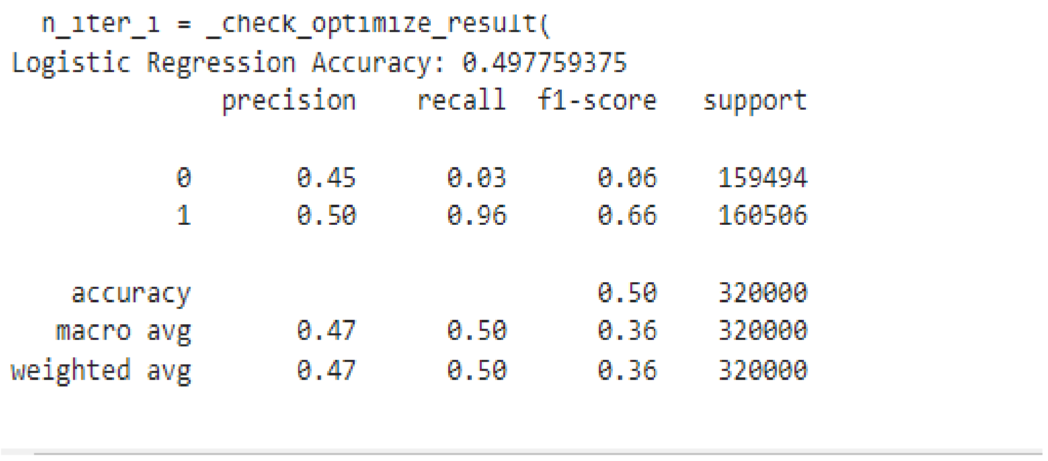
c. **Neutral Tweets Scenario**


..
sample_data
sentimentanalysis.h5
sentimentanalysis_encoder.pkl
sentimentanalysis_tokenizer.pkl

```
Enter text to analyze sentiment (or type 'exit' to quit): All right
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
Predicted Sentiment: NEUTRAL
Enter text to analyze sentiment (or type 'exit' to quit): It's okay
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
Predicted Sentiment: NEUTRAL
Enter text to analyze sentiment (or type 'exit' to quit): you'll figure out
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
Predicted Sentiment: NEUTRAL
Enter text to analyze sentiment (or type 'exit' to quit): Not bad, not great
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step
Predicted Sentiment: NEUTRAL
Enter text to analyze sentiment (or type 'exit' to quit): exit
Exiting the program.
```

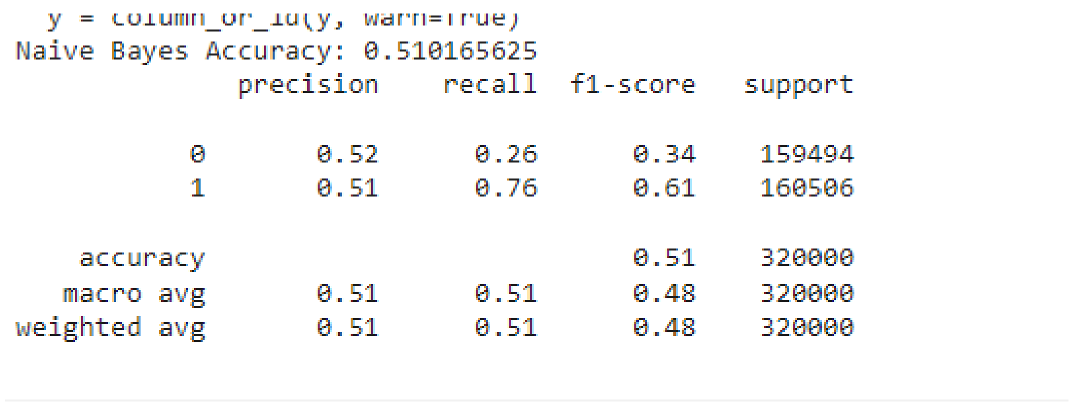
MODEL EVALUATION RESULTS

Logistic Regression



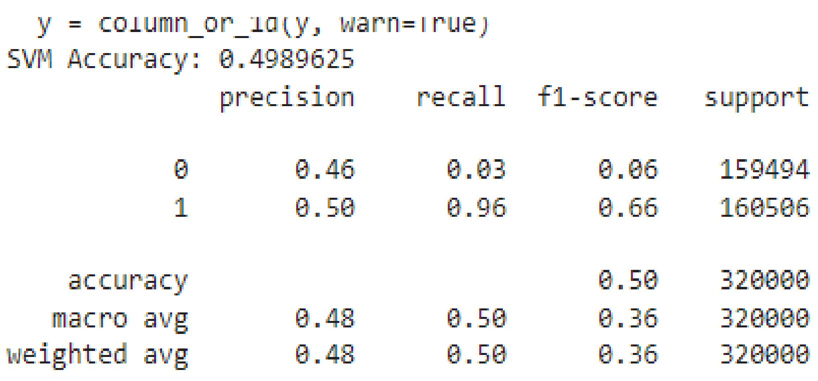
```
  n_iter_i = _check_optimize_result(
Logistic Regression Accuracy: 0.497759375
              precision    recall  f1-score   support

           0       0.45      0.03      0.06    159494
           1       0.50      0.96      0.66    160506

    accuracy                           0.50    320000
   macro avg       0.47      0.50      0.36    320000
weighted avg       0.47      0.50      0.36    320000
```

Naive Bayes



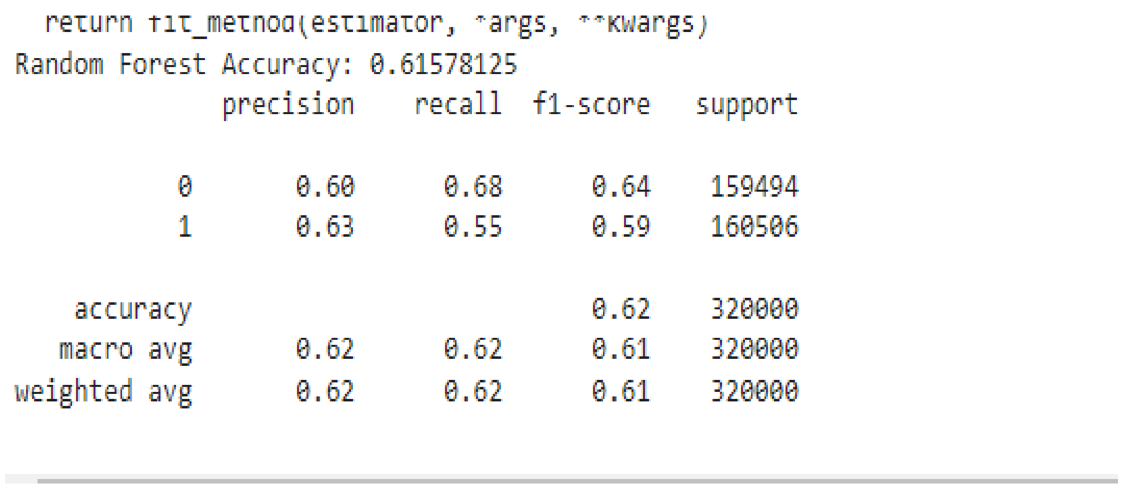
```
  y = column_or_1d(y, warn=True)
Naive Bayes Accuracy: 0.510165625
              precision    recall  f1-score   support

           0       0.52      0.26      0.34    159494
           1       0.51      0.76      0.61    160506

    accuracy                           0.51    320000
   macro avg       0.51      0.51      0.48    320000
weighted avg       0.51      0.51      0.48    320000
```

Support Vector Machines (SVM)



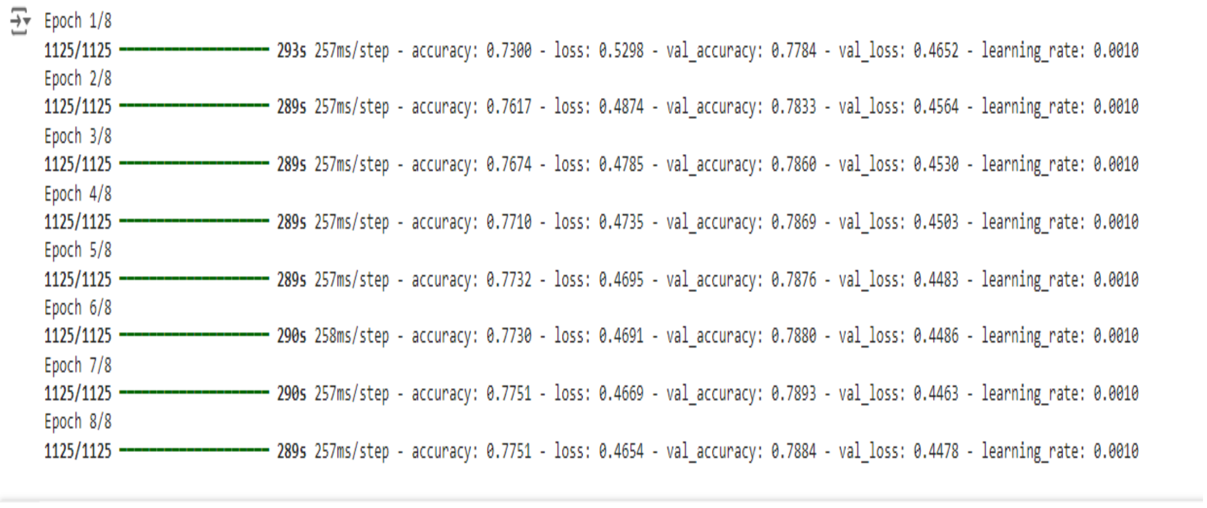
```
  y = column_or_1d(y, warn=True)
SVM Accuracy: 0.4989625
              precision    recall  f1-score   support

           0       0.46      0.03      0.06    159494
           1       0.50      0.96      0.66    160506

    accuracy                           0.50    320000
   macro avg       0.48      0.50      0.36    320000
weighted avg       0.48      0.50      0.36    320000
```

Random Forest



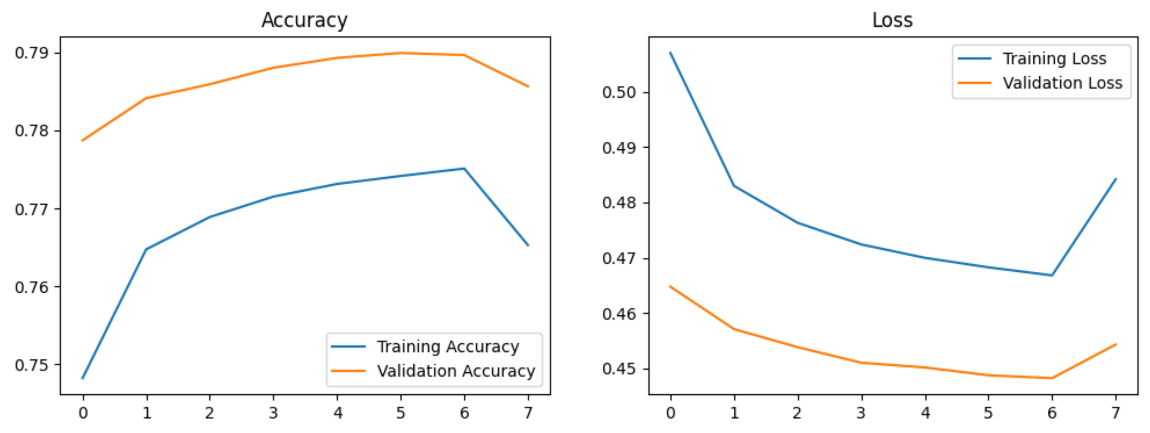
```
  return fit_method(estimator, *args, **kwargs)
Random Forest Accuracy: 0.61578125
              precision    recall  f1-score   support

           0       0.60      0.68      0.64    159494
           1       0.63      0.55      0.59    160506

    accuracy                           0.62    320000
   macro avg       0.62      0.62      0.61    320000
weighted avg       0.62      0.62      0.61    320000
```

Deep Learning models like LSTM (Long Short-Term Memory)



```
Epoch 1/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 293s 257ms/step - accuracy: 0.7300 - loss: 0.5298 - val_accuracy: 0.7784 - val_loss: 0.4652 - learning_rate: 0.0010
Epoch 2/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 289s 257ms/step - accuracy: 0.7617 - loss: 0.4874 - val_accuracy: 0.7833 - val_loss: 0.4564 - learning_rate: 0.0010
Epoch 3/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 289s 257ms/step - accuracy: 0.7674 - loss: 0.4785 - val_accuracy: 0.7860 - val_loss: 0.4530 - learning_rate: 0.0010
Epoch 4/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 289s 257ms/step - accuracy: 0.7710 - loss: 0.4735 - val_accuracy: 0.7869 - val_loss: 0.4503 - learning_rate: 0.0010
Epoch 5/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 289s 257ms/step - accuracy: 0.7732 - loss: 0.4695 - val_accuracy: 0.7876 - val_loss: 0.4483 - learning_rate: 0.0010
Epoch 6/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 290s 258ms/step - accuracy: 0.7730 - loss: 0.4691 - val_accuracy: 0.7880 - val_loss: 0.4486 - learning_rate: 0.0010
Epoch 7/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 290s 257ms/step - accuracy: 0.7751 - loss: 0.4669 - val_accuracy: 0.7893 - val_loss: 0.4463 - learning_rate: 0.0010
Epoch 8/8
1125/1125 ━━━━━━━━━━━━━━━━━━━━ 289s 257ms/step - accuracy: 0.7751 - loss: 0.4654 - val_accuracy: 0.7884 - val_loss: 0.4478 - learning_rate: 0.0010
```

Plot Accuracy and Loss Curves (LSTM)


Accuracy
Training Accuracy
Validation Accuracy
Loss
Training Loss
Validation Loss